

```

// Computer Program Listing Appendix Under 37 CFR 1.52(e)
// FEEmbed.java
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
/*
 * FEEmbed.java
 *
 *
 *
 */
package com.onepage.ccl.execute;
import java.io.*;
import java.util.*;
import java.net.*;
import com.onepage.html.parser.*;
import com.onepage.ccl.exceptions.CCLEException;
import com.onepage.ccl.exceptions.NoKeyFoundException;
import com.onepage.ccl.utils.ZHashtableNS;
/**
 * The Feature Key for an Embed is EMB0AAAAAANN where:<br>
 * "EMB" is the Feature Tag,<br>
 * '0' is the key version number, <br>
 * AAAAAA are the attributes and <br>
 * NNN is the number of occurrences from 001 to 999.<p>
 * <b> The Feature Attributes for this Feature Type are:</b><br>
 * <center><table>
 * <tr><td>RELEVANCE_INDEX</td><td>Rates the URL of the anchor and the protocol, server, source, banner,
etc</td></tr>
 * <tr><td>ADS_FLAG_INDEX</td><td>'Z' if the embedded object appears to be an advertisement. 'M'
otherwise.</td></tr>
 * <tr><td>HEIGHT_INDEX</td><td>Height attribute of the embed tag</td></tr>
 * <tr><td>WIDTH_INDEX</td><td>Width attribute of the embed tag</td></tr>
 * <tr><td>SRC_CHARS_INDEX</td><td>Characters in the src attribute</td></tr>
 * <tr><td>SRC_DEPTH_INDEX</td><td>Folder depth of the src attribute</td></tr>
 * </table></center>
 *
 */
public class FEEmbed implements FEInterface
{
    private static final char    VERSION = '0'; // TAG Attribute list version number
    // Feature Key array index definitions
    private static final int     FEATURE_TAG_INDEX1 = 0;    // 'E'
    private static final int     FEATURE_TAG_INDEX2 = 1;    // 'M'
    private static final int     FEATURE_TAG_INDEX3 = 2;    // 'B'
    private static final int     VERSION_INDEX = 3;    // version number
    private static final int     RELEVANCE_INDEX = 4;    // Rate protocol, server, source, banner, etc.
    private static final int     ADS_FLAG_INDEX = 5;    // URL contains 'ads' or Has 'click through' data after .gif
or .jpg
    private static final int     HEIGHT_INDEX = 6;    // Height attribute of the embed tag
    private static final int     WIDTH_INDEX = 7;    // Width attribute of the embed tag
    private static final int     SRC_CHARS_INDEX = 8;    // characters in the src file name

```

```

private static final int    SRC_DEPTH_INDEX =    9;    // Folder depth of the src file
// could add number of <param> tags
private static final int    FEATURE_KEY_SIZE =    10;        // Number of objects in a tag
public Vector _tableTags;
private CachedURL _cachedUrl;
private Hashtable _baseTagTable;
// constructors
public FEEmbed(URL theURL)
    throws MalformedURLException
{
    try
    {
        _cachedUrl = new CachedURL(theURL);
    }
    catch (MalformedURLException e)
    {
        System.out.println("FEGraphic::FEGraphic ERROR: Failed to create CachedURL due to \n" + e);
        throw e;
    }
}

public FEEmbed(String filename)
    throws MalformedURLException
{
    URL url = null;
    String lowerFilename = filename.toLowerCase();
    if (!lowerFilename.startsWith("http://") && !lowerFilename.startsWith("https://") &&
!lowerFilename.startsWith("file://"))
    {
        // need to check for local directory too!!! (for testing)
        filename = "http://" + filename;
    }
    // now create the URL
    try
    {
        url = new URL(filename);
        _cachedUrl = new CachedURL(url);
    }
    catch (MalformedURLException e)
    {
        System.out.println("FEButton::FEButton ERROR: Failed to create URL due to \n" + e);
        throw e;
    }
}

public FEEmbed(CachedURL theCachedURL)
{
    _cachedUrl = theCachedURL;
}

public FEEmbed(CachedURL theCachedURL, String versionStr)
{
    _cachedUrl = theCachedURL;
}

```

```

}
/**
 * tagToKey is called by the Navigation software and by the Capture Wizard to translate
 * an <embed> tag string into a Feature Extraction key.<P>
 * When the tag is returned it does not include the 'occurrence' count, which must be
 * maintained by the object calling FEEEmbed::tagToKey.<P>
 *
 * @param urlAndHTML The URL and HTML for the source page
 * @param embedTag The html <embed> tag string for this embedded object
 *
 * @return A feature key containing the feature tag "EMB", the version number '0', and
 * a list of rated attributes expressed as single capital letters.
 */
public static String tagToKey(CachedURL urlAndHtml, String embedTag)
    throws MalformedURLException
{
    //System.out.println("Entering FEEEmbed::tagToKey embedTag = " + embedTag);
    int srcCharsCount = 0; // characters in src file name
    int srcFolderCount = 0; // folder depth of the src
    char[] attributeValue = new char[FEATURE_KEY_SIZE]; // array of feature attributes
    // set feature type
    attributeValue[FEATURE_TAG_INDEX1] = 'E';
    attributeValue[FEATURE_TAG_INDEX2] = 'M';
    attributeValue[FEATURE_TAG_INDEX3] = 'B';
    // Set Version Number
    attributeValue[VERSION_INDEX] = VERSION;
    // initialize the rest of the attributes
    for (int i=4; i<FEATURE_KEY_SIZE; i++)
    {
        attributeValue[i] = 'M';
    }
    // PROCESS EMBED TAG
    //System.out.println("FEEEmbed::tagToKey about to process embed tag");
    ElementParser ep = new ElementParser();
    com.onepage.html.parser.Element elem = null;
    try
    {
        elem = (com.onepage.html.parser.Element) ep.parse(embedTag);
    }
    catch (java.io.IOException e)
    {
        return "badelement";
    }
    // process the src attribute
    //System.out.println("FEEEmbed::tagToKey before process src attribute");
    String srcAttribute = elem.getAttribute("src"); // make sure code is lowercase
    if (srcAttribute == null || srcAttribute.equals(""))
    {
        return "noCODEintag";
    }
}

```

```

else
{
    srcAttribute = srcAttribute.toLowerCase();
}
// Determine number of characters in the src file name
String fileName = srcAttribute.substring(srcAttribute.lastIndexOf('/')+1);
//System.out.println("FEEmbed::tagToKey fileName = " + fileName);
srcCharsCount = fileName.length();
// Determine folder depth of the src
for ( int i=0; i<srcAttribute.length()-4; i++ )
{
    // the depth will be inaccurate if the codebase is relative, but this is good enough for now
    // stretching the srcAttribute here would mess up the following relevance check anyway
    if (srcAttribute.charAt(i) == '/')
        srcFolderCount++;
}
//System.out.println("FEEmbed::tagToKey about to process attributes");
//
// ATTRIBUTE 1
// RELEVANCE_INDEX
//
if (srcAttribute.startsWith("http"))
{
    attributeValue[RELEVANCE_INDEX] = 'Z'; // probably offsite so add banner
    String strippedBase = "";
    if (urlAndHtml != null)
    {
        // note: toLowerCase causes a loss of capitalization data
        strippedBase = new String( urlAndHtml.GetURL().getHost()).toLowerCase();
    }
    // Strip www
    if( strippedBase.startsWith("www") )
        strippedBase = strippedBase.substring( 4);
    // Strip .org, .com, .net ...
    int dotCom = strippedBase.lastIndexOf( '.');
    if( dotCom!= -1)
        strippedBase = strippedBase.substring( 0, dotCom);
    // If the referral contains similar names, do not be so harsh on the rating
    if( srcAttribute.indexOf( strippedBase) != -1)
        attributeValue[RELEVANCE_INDEX] = 'M'; // Not as good as local
        // better than none
}
else
{
    attributeValue[RELEVANCE_INDEX] = 'M'; /// no clue so mid-range value
}
//
// ATTRIBUTE 2
// ADS_FLAGS_INDEX
//

```

```

// Look for 'ads' in URL
attributeValue[ADS_FLAG_INDEX] = 'M';
// note: codebaseAttribute and codeAttribute are made lowercase above (so we don't need indexIgnoreCaseOf
here)
if (srcAttribute.indexOf("graphics") > 0)
{
    attributeValue[ADS_FLAG_INDEX] = 'X';
}
if (srcAttribute.indexOf("ads.") > 0 || srcAttribute.indexOf("ads/") > 0)
{
    attributeValue[ADS_FLAG_INDEX] = 'Z';
}
//
// ATTRIBUTE 3
// HEIGHT_INDEX - height attribute value
//
String heightAttribute = elem.getAttribute("height");
int h = FESStatic.GetInit(heightAttribute);
attributeValue[HEIGHT_INDEX] = 'M';
if (h < 20)
    attributeValue[HEIGHT_INDEX] = 'Z';
if (h >= 20 && h < 49)
    attributeValue[HEIGHT_INDEX] = 'F';
else if (h >= 49 && h < 99)
    attributeValue[HEIGHT_INDEX] = 'E';
else if (h >= 99 && h < 149)
    attributeValue[HEIGHT_INDEX] = 'D';
else if (h >= 149 && h < 199)
    attributeValue[HEIGHT_INDEX] = 'C';
else if (h >= 199 && h < 299)
    attributeValue[HEIGHT_INDEX] = 'B';
else if (h >= 299)
    attributeValue[HEIGHT_INDEX] = 'A';
//
// ATTRIBUTE 4
// WIDTH_INDEX - width attribute value
//
String widthAttribute = elem.getAttribute("width");
int w = FESStatic.GetInit(widthAttribute);
attributeValue[WIDTH_INDEX] = 'M';
if (w < 20)
    attributeValue[WIDTH_INDEX] = 'Z';
if (w >= 20 && w < 49)
    attributeValue[WIDTH_INDEX] = 'F';
else if (w >= 49 && w < 99)
    attributeValue[WIDTH_INDEX] = 'E';
else if (w >= 99 && w < 149)
    attributeValue[WIDTH_INDEX] = 'D';
else if (w >= 149 && w < 199)
    attributeValue[WIDTH_INDEX] = 'C';

```

```

else if (w >= 199 && w < 299)
    attributeValue[WIDTH_INDEX] = 'B';
else if (w >= 299)
    attributeValue[WIDTH_INDEX] = 'A';
//
// ATTRIBUTE 5
// SRC_CHARS_INDEX - Length of the src file name
//
if (srcCharsCount < 26)
    attributeValue[SRC_CHARS_INDEX] = (char) ('Z' - srcCharsCount);
else
    attributeValue[SRC_CHARS_INDEX] = 'A';
//
// ATTRIBUTE 6
// kFolderDepthAttribute - folder depth of image file
//
if (srcFolderCount < 26)
    attributeValue[SRC_DEPTH_INDEX] = (char) ('Z' - srcFolderCount);
else
    attributeValue[SRC_DEPTH_INDEX] = 'A';
// build key string from individual key attributes
String key = "";
for (int i=0; i<FEATURE_KEY_SIZE; i++)
    key += attributeValue[i];
//System.out.println("Leaving FEEmbed::tagToKey key = " + key);
return key;
}
/**
 * getContainer is a key routine that is passed the 'inside object' and returns the tag
 * of its 'container'. In the preview window this routine gives the ability to zoom out
 * from an inside object.
 *
 * @param    insideTag the feature tag of the inside object.
 * @return
 *
 */
public String getContainer(String insideTag)
{
    //System.out.println("Entering FECore::getContainer insideTag = " + insideTag);
    getFEStates();
    String tag;
    FEStateAbstract state;
    if (FEStatic.isItTagOfType( insideTag, FEConstants.STR_FEATURE_TAG_APPLET))
    {
        for (int i = 0; i < _tableTags.size(); i++)
        {
            state = (FEStateAbstract) _tableTags.elementAt(i);
            tag = state.findTag(insideTag);
            if (!tag.equals(""))
            {

```

```

        return tag;
    }
}
return "";
}
public Vector getFEStates()
{
    if (_tableTags == null)
    {
        try
        {
            String text = _cachedUrl.GetHTMLCachedData();
            String urlStr = _cachedUrl.GetURL().toString();
            FEEmbedParser parser = new FEEmbedParser(_cachedUrl);
            parser.parse(urlStr, text, null, -1, true, null, 0);
            _tableTags = parser.getTableTags();
            _baseTagTable = parser.getBaseTags();
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return new Vector();
        }
    }
    return _tableTags;
}
/**
 * buildPreview is passed the attribute key for an Embed contained in a page and will
 * construct a HTML string that will be used by the preview servlet OCview.
 *
 * @param fullKey The feature key, originally generated by tagToKey above, that will be
 * used to find the appropriate stored EMBED tag and construct the embedded object that we wish
 * to display.
 *
 * @return An HTML string containing the <embed> tag string that will display the
 * embedded object that we're looking for.
 */
public String buildPreview(String fullKey, DataBubble bubble)
{
    //System.out.println("Entering FEEmbed::buildPreview fullKey = " + fullKey);
    FEStateAbstract theState;
    String embedTag;
    StringBuffer sb = new StringBuffer();
    sb.append("<!-- Begin FEEmbed Capture -->");
    if ( FEStatic.isItTagOfType( fullKey, FEConstants.STR_FEATURE_TAG_EMBED ) )
    {
        getFEStates();
        int i = 0;
        while (i < _tableTags.size())

```

```

{
theState = (FESStateAbstract)_tableTags.elementAt(i++);
if (theState != null)
{
    embedTag = theState.getElementWithKey(fullKey);
    //System.out.println("FEEmbed::buildPreview original imgTag = " + imgTag);
    if (embedTag.length() > 2) // make sure a decent embed tag was returned
    {
        // set up base URL
        BaseTag baseTag = (BaseTag)_baseTagTable.get(fullKey);
        URL baseUrl = null;
        try
        {
            if(baseTag != null)
            {
                baseUrl = new URL(baseTag.href());
            }
            else
            {
                System.out.println("FEEmbed::buildPreview ERROR: Failed to retrieve feature key from
_baseTagTable.");
                baseUrl = _cachedUrl.GetURL();
            }
        }
        catch(java.net.MalformedURLException e)
        {
            System.out.println("FEEmbed::buildPreview Could not create base URL from baseTag.");
            // note: a null URL will cause an exception later
        }
        // PARSE THE EMBED TAG AND PUT THE RESULTS IN "ELEM"
        ElementParser ep = new ElementParser();
        com.onepage.html.parser.Element elem = null;
        try
        {
            elem = (com.onepage.html.parser.Element) ep.parse(embedTag);
        }
        catch (java.io.IOException e)
        {
            return "badelement";
        }
        // MAKE SURE AUTOPLAY IS TURNED OFF IN PREVIEW MODE
        String autoPlay = elem.getAttribute("autoplay");
        if(autoPlay != null)
        {
            elem.setAttribute("autoplay","FALSE");
        }
        // ADD BASE URL INFORMATION INTO SRC ATTRIBUTE
        String srcAttribute = elem.getAttribute("src");
        if (srcAttribute == null || srcAttribute.equals(""))
        {

```



```

        // note: for embed tags, a missing src is valid. (uses MIME type to load plugin)
        // so, we don't need to add a src attribute if it is missing.
    }
    else
    {
        // make sure existing src is absolute (fully qualified)
        URL newURL = null;
        try{
            //newURL = new URL(_cachedUrl.GetURL(), srcAttribute);
            newURL = new URL(baseUrl, srcAttribute);
        }
        catch(java.net.MalformedURLException e)
        {
            // handle exception here
            System.out.println("FEEmbed::buildPreview Could not create URL for src.");
        }
        srcAttribute = newURL.toString();
        elem.setAttribute("src",srcAttribute);
    }
    // END ADD BASE URL INFORMATION INTO SRC ATTRIBUTE
    // get embed tag back out of elem and add to output buffer
    sb.append(elem.toHtmlString());
    sb.append("<!-- End FEEmbed Capture -->");
    //System.out.println("Leaving FEEmbed::buildPreview sb = " + sb.toString());
    return sb.toString();
}
}
}
}
//return "Embed not found";
System.out.println("FEEmbed::buildPreview ERROR: Embed Not Found!");
return "";
}
/**
 * buildFinal is passed the attribute key for an embedded object contained in a page and will
 * construct a HTML string that will be used by CCL_FEATURE and by
 * TileCCL when the Button is rendered on the user's page.
 *
 * @param fullKey The feature key, originally generated by tagToKey above, that will be
 * used to find the appropriate stored embed tag and construct the embedded object that we wish
 * to display.
 *
 * @return An HTML string containing the embed tag that will display the embedded object
 * that we're looking for.
 */
public String buildFinal(String fullKey, DataBubble bubble)
{
    //System.out.println("Entering FEEmbed::buildFinal fullKey = " + fullKey);
    FEStateAbstract theState;
    String embedTag;

```

```

StringBuffer sb = new StringBuffer();
sb.append("<!-- Begin FEEmbed Capture -->");
if ( FEStatic.isItTagOfType( fullKey, FEConstants.STR_FEATURE_TAG_EMBED ) )
{
    getFEStates();
    int i = 0;
    while (i < _tableTags.size())
    {
        theState = (FEStateAbstract)_tableTags.elementAt(i++);
        if (theState != null)
        {
            embedTag = theState.getElementWithKey(fullKey);
            //System.out.println("FEEmbed::buildPreview original imgTag = " + imgTag);
            if (embedTag.length() > 2) // make sure a decent embed tag was returned
            {
                // set up base URL
                BaseTag baseTag = (BaseTag)_baseTagTable.get(fullKey);
                URL baseURL = null;
                try
                {
                    if(baseTag != null)
                    {
                        baseURL = new URL(baseTag.href());
                    }
                    else
                    {
                        System.out.println("FEEmbed::buildPreview ERROR: Failed to retrieve feature key from
_baseTagTable.");
                        baseURL = _cachedUrl.GetURL();
                    }
                }
                catch(java.net.MalformedURLException e)
                {
                    System.out.println("FEEmbed::buildPreview Could not create base URL from baseTag.");
                    // note: a null URL will cause an exception later
                }
                // PARSE THE EMBED TAG AND PUT THE RESULTS IN "ELEM"
                ElementParser ep = new ElementParser();
                com.onepage.html.parser.Element elem = null;
                try
                {
                    elem = (com.onepage.html.parser.Element) ep.parse(embedTag);
                }
                catch (java.io.IOException e)
                {
                    return "badelement";
                }
                // ADD BASE URL INFORMATION INTO SRC ATTRIBUTE
                String srcAttribute = elem.getAttribute("src");
                if (srcAttribute == null || srcAttribute.equals(""))

```

```

    {
        // note: for embed tags, a missing src is valid. (uses MIME type to load plugin)
        // so, we don't need to add a src attribute if it is missing.
    }
    else
    {
        // make sure existing src is absolute (fully qualified)
        URL newURL = null;
        try{
            //newURL = new URL(_cachedUrl.GetURL(), srcAttribute);
            newURL = new URL(baseUrl, srcAttribute);
        }
        catch(java.net.MalformedURLException e)
        {
            // handle exception here
            System.out.println("FEEmbed::buildFinal Could not create URL for src.");
        }
        srcAttribute = newURL.toString();
        elem.setAttribute("src",srcAttribute);
    }
    // END ADD BASE URL INFORMATION INTO SRC ATTRIBUTE
    // get embed tag back out of elem and add to output buffer
    sb.append(elem.toHtmlString());
    // add closing tag to output buffer ( for XHTML )
    sb.append("</embed>");
    sb.append("<!-- End FEEmbed Capture -->");
    //System.out.println("Leaving FEEmbed::buildPreview sb = " + sb.toString());
    return sb.toString();
}
}
}
}
//return "Embed not found";
System.out.println("FEEmbed::buildFinal ERROR: Embed Not Found!");
return "";
}
/**
 * partialHtmlToKey returns the feature tag asociated with the html argument
 *
 * @param    partialHtml some html text
 *
 * @return    the key asociated with the html argument
 */
public String partialHtmlToKey(String partialHtml)
    throws CCLErrorException, IllegalArgumentException
{
    if (partialHtml == null || partialHtml.equals(""))
    {
        throw new IllegalArgumentException("argument is empty");
    }
}

```

```

getFEStates();
// loop thru keys for each state
for (int j = 0; j < _tableTags.size(); j++)
{
    FEStateAbstract state = (FEStateAbstract)_tableTags.elementAt(j);
    Hashtable ht = state.getStateKeys();
    Enumeration emkeys = ht.keys();
    while (emkeys.hasMoreElements()){
        String key = (String)emkeys.nextElement();
        if ( FEStatic.isItTagOfType( key, FEConstants.STR_FEATURE_TAG_EMBED ) ) {
            String value = (String)ht.get(key);
            if (value.indexOf(partialHtml) != -1)
            {
                // found the key so return
                return key;
            }
        }
    } // while
}
// did not find a matched key for the html
throw new NoKeyFoundException();
}
/**
 * getAllKeys is overwritten by each FE class to return a vector containing the keys for all of
 * the features found.
 *
 * @return vector containing all of the keys
 */
public Vector getAllKeys()
    throws CCLException, IllegalArgumentException
{
    getFEStates();
    Vector allKeys = new Vector();
    // loop thru keys for each state
    for (int j = 0; j < _tableTags.size(); j++)
    {
        FEStateAbstract state = (FEStateAbstract)_tableTags.elementAt(j);
        Hashtable ht = state.getStateKeys();
        Enumeration emkeys = ht.keys();
        while (emkeys.hasMoreElements()){
            String key = (String)emkeys.nextElement();
            if ( FEStatic.isItTagOfType( key, FEConstants.STR_FEATURE_TAG_EMBED ) ) {
                allKeys.addElement(key);
            }
        } // while
    }
    return allKeys;
}
/**
 * getAllFeatures is overwritten by each FE class to return a vector containing all of the

```

```

* features found.
*
* @return vector containing all of the features
*/
public Vector getAllFeatures()
    throws CCLException, IllegalArgumentException
{
    getFEStates();
    Vector allFeatures = new Vector();
    // loop thru keys for each state
    for (int j = 0; j < _tableTags.size(); j++)
    {
        FEStateAbstract state = (FEStateAbstract)_tableTags.elementAt(j);
        Hashtable ht = state.getStateKeys();
        Enumeration emkeys = ht.keys();
        while (emkeys.hasMoreElements()){
            String key = (String)emkeys.nextElement();
            if ( FEStatic.isItTagOfType( key, FEConstants.STR_FEATURE_TAG_EMBED ) ) {
                String value = (String)ht.get(key);
                allFeatures.addElement(value);
            }
        } // while
    }
    return allFeatures;
}

public String buildPreviewToXML(String key, DataBubble bubble) throws Exception
{
    return(FEConstants.NO_VALID_XML);
}

public String buildFinalToXML(String key, DataBubble bubble) throws Exception
{
    return(FEConstants.NO_VALID_XML);
}

protected void hdout(char c)
{
}

protected void hdout(String s)
{
}
}

```